

# **FORTRAN Implementation of Tutorial Input**

K. Moyd

Communications Systems Research Section

*This article describes the FORTRAN implementation of "Tutorial Input," a computer/human interface for real-time control programs. The emphasis is on the communication between the standardized input routine and a real-time FORTRAN control program. Changes made to the Tutorial Input specifications are explained, and samples of the use of this implementation are given.*

## **I. Introduction**

"Tutorial Input" is a standardized computer/human interface for real-time control programs. It was developed by A. I. Zygielbaum (Ref. 1), who has implemented an assembly language version for the PDP 11/20. A version has been written in real-time FORTRAN II for the XDS 930 computer, to be used in the pulsar automation demonstration. Some additions were made to the Tutorial Input specifications to increase the interface flexibility; some modifications were made to permit the input routine to be written entirely in FORTRAN. Much of the input routine could be used for other computers and projects.

A summary of Tutorial Input is given in Section II; the communication between the control program and the input routine is described in Section III; the changes to the original specifications are summarized in Section IV; and examples of the use of Tutorial Input are given in Section V. A description of the Tutorial Input routine and the flowcharts are presented in the Appendix.

## **II. Summary of Tutorial Input**

Tutorial Input is a method by which an operator enters parameters and control information into a real-time control program written by a user. It allows the operator to determine how much help he needs in entering information rather than having a completely preprogrammed series of messages or requiring him to memorize the entry sequence. The input is divided into commands and parameters. A command is a name associated with a specific set of numeric and/or alphanumeric parameters. A control command does not have any parameters. Once the operator enters a command, he may enter as many of its parameters in order as he knows. If one or more parameters are not entered, the computer will prompt the operator by typing out the name of the next parameter. The process continues until all the parameters are entered.

If the operator determines that he made an error, he may cancel the input, causing an abnormal exit from the

input routine; he may backspace over the incorrect character(s); he may delete the entire line; or he may retype the incorrect command. An asterisk typed in place of a parameter value causes the previously accepted value of that parameter to be used. Certain errors are detected by the computer. An error indication is typed out, and operator correction is expected. The input routine is terminated normally when no new command follows on the same line as the last parameter of the previous command, if there have been no uncorrected errors.

### III. Interfacing with the User's Program

All communication between the user's program and the input routine is by means of tables in COMMON. These tables must be initialized by the user before the input routine is entered the first time. The names used below to define the tables are those used in the input routine.

A set of four tables defines the commands, and a set of three tables defines the parameters. Each command table must be dimensioned at least NCOM (in COMMON), the number of commands. The ordering of commands is not important. ICOM(I) gives the four-character name of the I<sup>th</sup> command. INPAR(I) is the number of parameters, and INDEX(I) is the index in the parameter tables of the first parameter associated with that command. IFLAG(I) is set to 1 upon normal exit from the input routine if the I<sup>th</sup> command was successfully entered. Each parameter table must be dimensioned at least NPAR (in COMMON), the total number of parameters. The parameters for the I<sup>th</sup> command have indices from INDEX(I) to INDEX(I) + INPAR(I) - 1. One parameter may be included in several commands. Control commands (no parameters) have no entries in the parameter tables. INAME(J) is the four-character name typed out if the J<sup>th</sup> parameter is the next one to be entered. ICODE(J) specifies how the parameter is to be decoded. In this version ICODE(J) = -1 for floating point, 0 for integer, and 1 for alphanumeric. Because floating point numbers take two words on the XDS 930 while integers take only one, there are two tables for returning parameter values, PARAM and IPARAM. IPIND(J) is the index in the appropriate parameter storage table; ICODE(J) determines which table is to be used. Alphanumeric parameters are stored in the integer table and are therefore limited to four characters on the XDS 930.

Both the flag table (IFLAG) and the parameter storage tables are buffered in the input routine. The internal parameter buffer is entirely floating point. The values in the parameter storage tables are moved to the internal parameter buffer upon entry to the input routine, and the

updated values are moved from the internal buffer to the storage tables upon normal exit. Therefore, any change made in the parameter storage tables by the user's program while the input routine is active may be overwritten when the input routine is done. The flag table is buffered differently. An internal buffer is zeroed upon entry to the input routine. As each command is successfully processed, the corresponding flag is set in the buffer. Upon normal exit from the input routine, those flags corresponding to new input are set in the flag table. The other flags are not affected at all. Therefore, the user may safely reset flags even when the input routine is active.

The use of the parameters and flags is left to the user. In cases in which parameter changes may occur at any time, the value in the parameter storage table may be used as the parameter value in the user's program. In cases in which the timing of parameter changes is critical, the appropriate flag(s) can be checked and the values transferred from the storage tables to the active locations at the proper time. The flags corresponding to control commands can be tested and acted upon in a background loop or in appropriate interrupt routines. A flag should be reset as soon as its command is acted upon to decrease the probability of not recognizing a new set of parameters.

At the present time, the user determines how entry to the input routine is initiated. Two possible methods are console interrupt and breakpoint control. The user must make sure that operator input/output (I/O) does not interfere with any other I/O. This may be a problem if either the operator I/O or other I/O is interrupt controlled.

### IV. Additions and Modifications

The original Tutorial Input did not contain the flag table. Its inclusion allows parameter changes to be detected more easily and permits the timing of operator input to be made independent of the basic user program (with the exception of I/O interference). By means of the flag table, commands with no parameters can be used for program control. The only effect of such a control command is to set the corresponding flag.

The remaining tables are basically the same as in the assembly language implementation except that array indices are used in place of addresses. The internal parameter buffer is entirely floating point.

A method for indicating errors was established. Because a new line is not accepted until the previous line is

completely processed, any recognizable error must be on the last line typed. A  $\Delta$  on the line below marks the first character of the command or parameter in error. All commands and parameters preceding the  $\Delta$  have been accepted by the input routine; everything following the  $\Delta$  will be ignored. In the case of a command error, input is to be continued with the typing of a new command. In case of a parameter error, the message corresponding to that parameter is typed out, and input is to be continued with the entering of that parameter. An error message is typed on the line following the error indicator. The possible messages are:

**ILLEGAL COMMAND**—longer than four characters.

**UNRECOGNIZED COMMAND**—not in the table of command names.

**ILLEGAL PARAMETER**—longer than four characters for an alphanumeric parameter, illegal character for a numeric parameter.

The control characters  $R^c$ ,  $C^c$  and  $E^c$  described in Ref. 1 do not exist on the 930 typewriters. They have been replaced by  $\S$ ,  $<$ , and  $\#\#$ , respectively. The  $\S$  causes immediate exit from the input routine without transferring any parameters from the internal buffer to the storage tables and without changing any flags. The  $<$  causes the previous character to be deleted. As many characters will be deleted as there are  $<$ s until the beginning of the line is reached. Excess  $<$ s will be ignored. The  $\#\#$  causes the entire line to be deleted. A carriage return may be done before retyping the line.

## V. Sample of Tutorial Input

A program was written to test the Tutorial Input routine. It set up the necessary tables in COMMON. The

command and parameter definitions are given in Table 1. In this case,  $NCOM = 4$  and  $NPAR = 5$ . Three of the parameters are stored in the integer table IPARAM; two, in the floating point table PARAM. Both tables were zeroed at the beginning of the test program.

The input routine was called whenever a specific breakpoint was set. The flag table was zeroed before each call. Upon return to the main program, the tables IFLAG, IPARAM, and PARAM were typed out.

Typewriter output from the test program is shown in Figs. 1 and 2. The entry numbers were added later. The underlined characters were typed out by the computer; the rest were entered by the operator. Figure 1 gives examples of normal use of Tutorial Input. Entries 1, 2, and 3 show the command POSN being entered in three ways: with no prompting, with complete prompting, and with partial prompting. The use of the asterisk, the effect of a no-parameter command (STOW), and the combining of several commands are shown in entries 4, 5, and 6, respectively. Figure 2 gives examples of the error indications and shows the use of the operator correction features. The parameters at the beginning of the generation of Fig. 2 were the same as at the end of Figure 1. Entry 1 shows two illegal commands (more than four characters). Entry 2 shows a command, STW, that is not in the command list. In this case, the parameters for RECV were accepted. Entry 3 shows an illegal parameter (a letter in a numeric parameter). In this case, the first parameter for RECV was accepted; only the second parameter (ATTN) had to be entered. The use of the input cancellation ( $\S$ ), backspace ( $<$ ), and line delete ( $\#\#$ ) are shown in entries 4, 5, and 6, respectively. In entry 6, a carriage return was typed before the line was reentered.

## Reference

1. Zygielbaum, A. I., "Tutorial Input—Standardizing the Computer/Human Interface", in *The Deep Space Network Progress Report 42-23*, pp. 78-86, Jet Propulsion Laboratory, Pasadena, California, October 15, 1974.

**Table 1. Tutorial input command and parameter definitions**

Command definitions			
Name	Number of parameters	Index of first parameter	
POSN	3	1	
TMCN	1	4	
STOW	0	—	
RECV	2	4	
Parameter definitions			
Index	Message	Decoding type	Storage index
1	RA	0 (integer)	1
2	DEC	0	2
3	TYPE	1 (alphanumeric)	3
4	TMCT	−1 (floating point)	1
5	ATTN	−1	2

<u>Entry</u>									
1	POSN/25,-56,A300	1	0	0	0	25	-56	A300	.00 .00
2	POSN/ <u>RA :</u> -98 <u>DEC :</u> 37 <u>TYPE:</u> ALPH	1	0	0	0	-98	37	ALPH	.00 .00
3	POSN/67,36 <u>TYPE:</u> NUM	1	0	0	0	67	36	NUM	.00 .00
4	POSN/*,49,A <u>STOW/</u>	1	0	0	0	67	49	A	.00 .00
		0	0	1	0	67	49	A	.00 .00
5	POSN/99,56,B,RECV/15.67,-38.92	1	0	0	1	99	56	B	15.67 -38.92
6	<u>STOW/RECV/</u> <u>TMCT:</u> 97.5 <u>ATTN:</u> 28.6	0	0	1	1	99	56	B	97.50 28.60

Fig. 1. Normal Input

1	STOW/ <u>Δ</u> <u>ILLEGAL COMMAND</u> STOW/RECV/69.5,37.9 ----- <u>Δ</u> <u>ILLEGAL COMMAND</u> RECV/*,58.5 0 0 1 1            99    56   B       97.50   58.50
2	RECV/28,-56,STW/ ----- <u>Δ</u> <u>UNRECOGNIZED COMMAND</u> STOW/ 0 0 1 1            99    56   B       28.00 -56.00
3	RECV/97.3,5A ----- <u>Δ</u> <u>ILLEGAL PARAMETER</u> ATTN: 365.7 0 0 0 1            99    56   B       97.30 365.70
4	STOW/RECV/6T# 0 0 0 0            99    56   B       97.30 365.70
5	STOW/RECV/6T<5.7,17 0 0 1 1            99    56   B       55.70   17.00
6	STOW/RECV/6T# STOW/RECV/63.7,59 0 0 1 1            99    56   B       63.70   59.00

Fig. 2. Errors

## Appendix

The Tutorial Input routine consists of a main routine named TUTOR and four subroutines named TYPEIN, AFIELD, NFIELD, and ERRIND. There is no additional COMMON needed for communication among these routines.

TYPEIN accepts one line of input, edits out backspaces, counts the resulting number of input characters, and sets a flag if the input cancellation character was entered. The input line is stored in an array with one character per word.

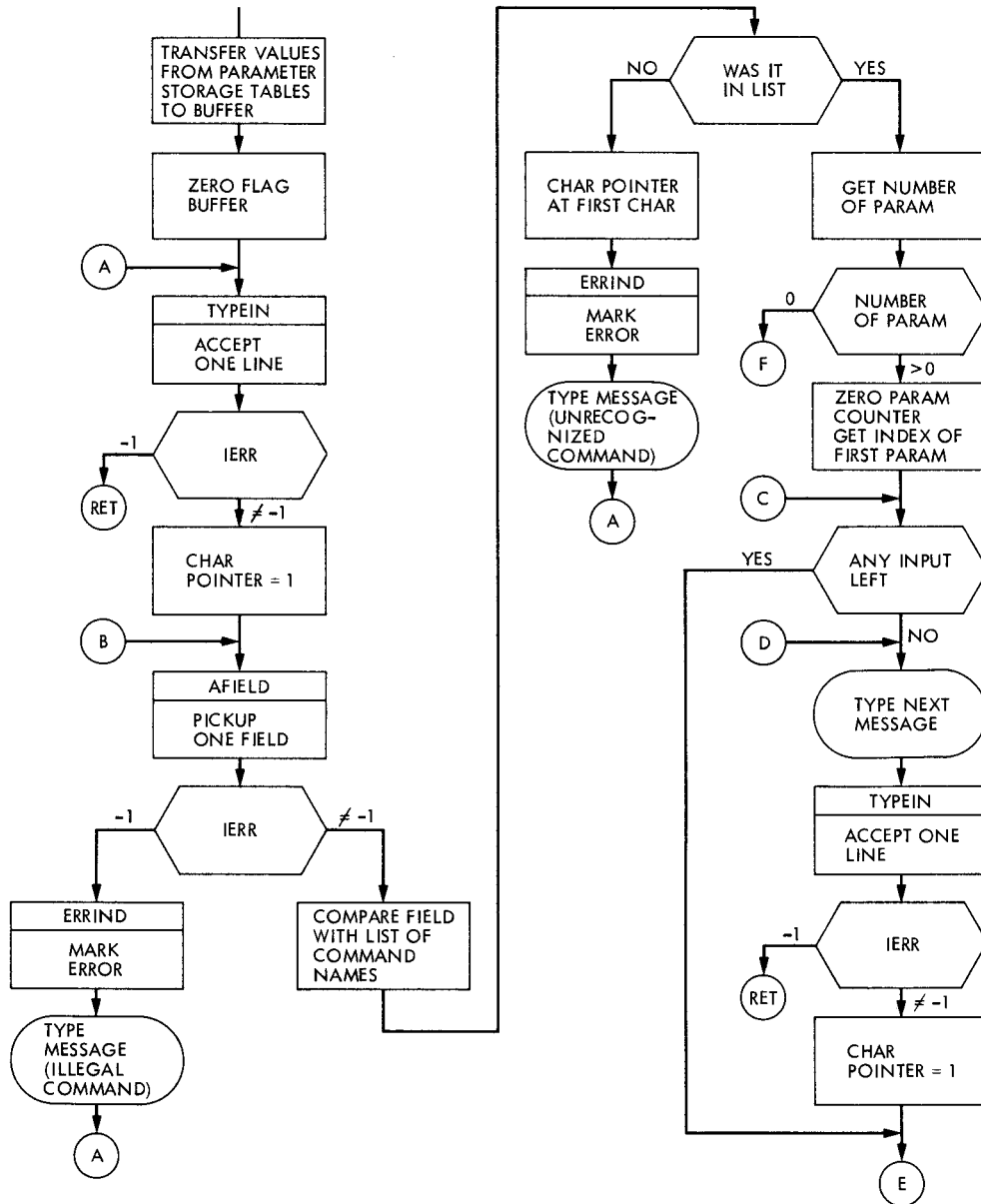
AFIELD takes the next field of the input line, terminated by a slash, comma, or blank (end of line), and packs it into one four-character word (blank filled to the right if necessary). The terminator is not included in the resulting word. If there are more than four characters, a flag is set to -1. If an asterisk occurs in the field, the flag is set to 1. Otherwise, the flag is 0.

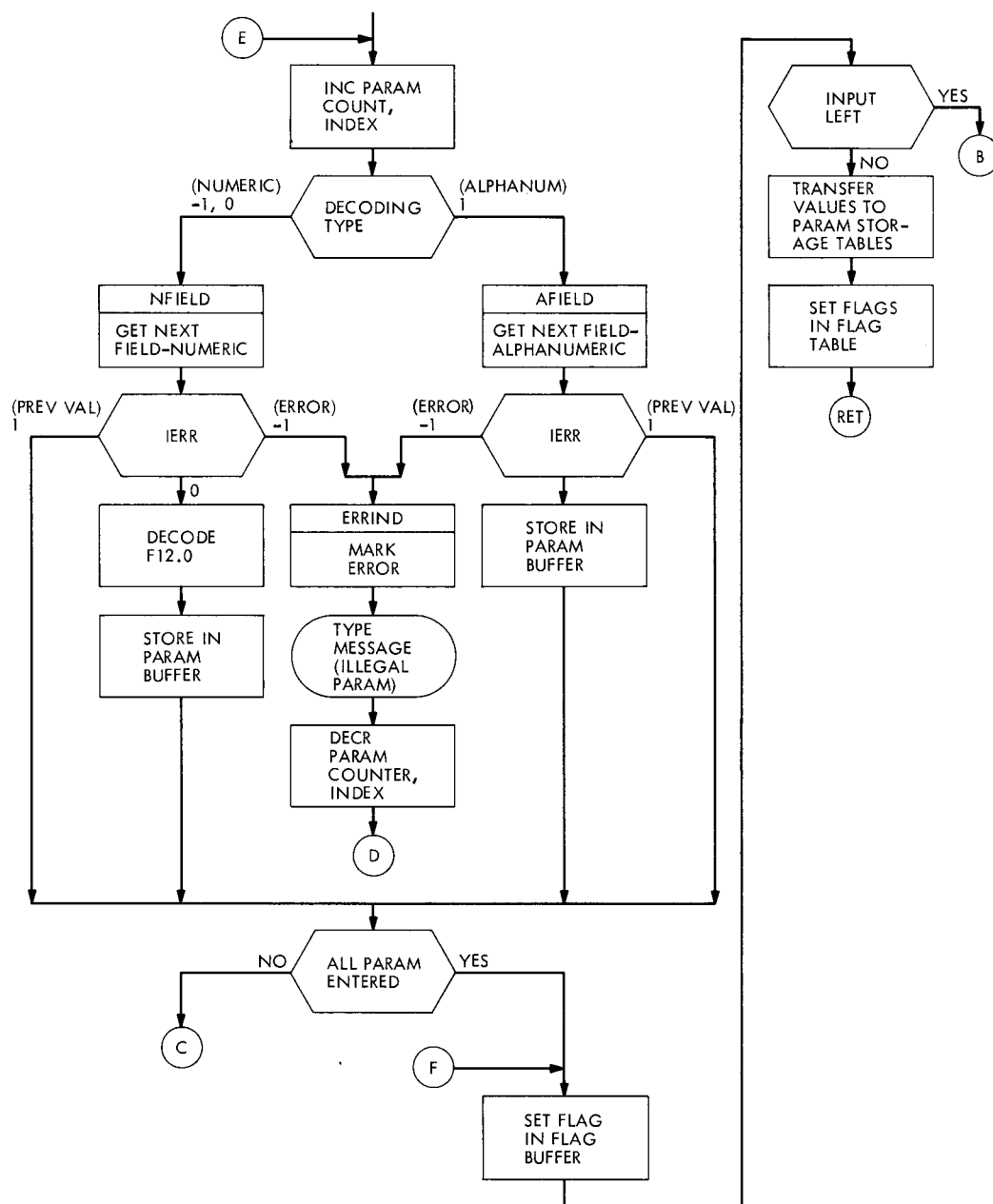
NFIELD takes the next field of the input line, terminated by a comma or blank, and packs it into three words, four characters per word (blank filled at the end if necessary). A comma is included as the last non-blank character if there are fewer than 12 characters. A flag is set to -1 if there are more than 12 characters or if there is a character other than an asterisk, number, sign, or decimal point. If an asterisk occurs in the field, the flag is set to 1. Otherwise the flag is 0.

ERRIND causes the error indicator to be typed out. It sets the indicator at the actual point of the error, even if the line included backspaces.

TUTOR takes care of the entire input procedure, including the accepting of input, processing of commands and parameters, error indications, and buffer transfers. The detailed procedures have been covered in previous sections of the article and will not be repeated here.

TUTOR





# AFIELD

